CARASERVE: CPU-Assisted and Rank-Aware LoRA Serving for Generative LLM Inference

基本信息

作者：

Suyi Li, Hanfeng Lu, Tianyuan Wu, Minchen Yu·, Qizhen Weng‡, Xusheng Chen†, Yizhou Shan†, Binhang Yuan, Wei Wang

HKUST, CUHK-Shenzhen, ‡Shanghai AI Laboratory, †Huawei Cloud

Introduction

多租户 LoRA 服务挑战：

- 在多租户云环境中，为每个 LoRA 适配器复制base model会浪费 GPU 内存且无法批量推理；
- 仅在 GPU 上缓存base model并按需加载 LoRA 适配器虽节省内存，但会导致冷启动问题，影响服务响应速度；
- 现有系统对异构 LoRA 请求调度缺乏优化，无法满足 SLO。

Continuous batching

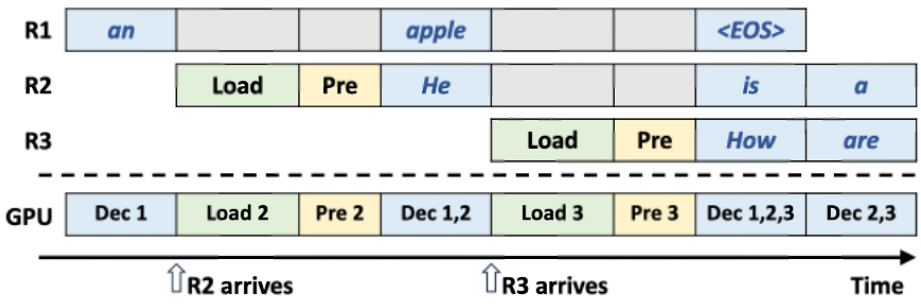


Figure 2: Continuous batching in which the decoding phase (Dec) is preempted to perform prompt processing upon a request arrival, which involves loading the requested LoRA adapter (Load) and prefilling (Pre).

当前新请求到达时，先中断解码阶段，对新请求进行load adapter 以及 prefill，完成后新请求与原来正在解码的请求组合在一起进行解码。

设计目标

	GPU-Efficient	Cold-Start-Free	SLO-Aware
HF-PEFT [14]	✗	✓	✗
S-LoRA [23]	✓	✗	✗
Punica [1]	✓	✗	✗
CARASERVE	✓	✓	✓

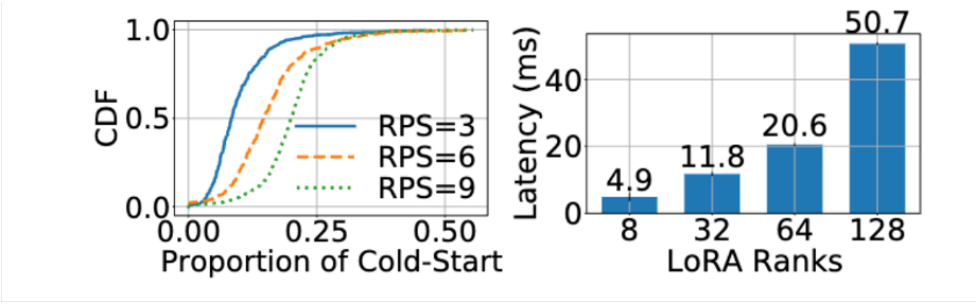
Table 1: Summarization of LoRA serving systems.

Challenge

Challenge1 高冷启动开销

Adapter存储在主存中，当新请求到达时，系统需要将其加载到GPU中，导致了冷启动问题

采用continuois batching的话，正在解码的请求会因为新请求的到来被阻塞，新请求的冷启动会延迟正在处理请求的token生成



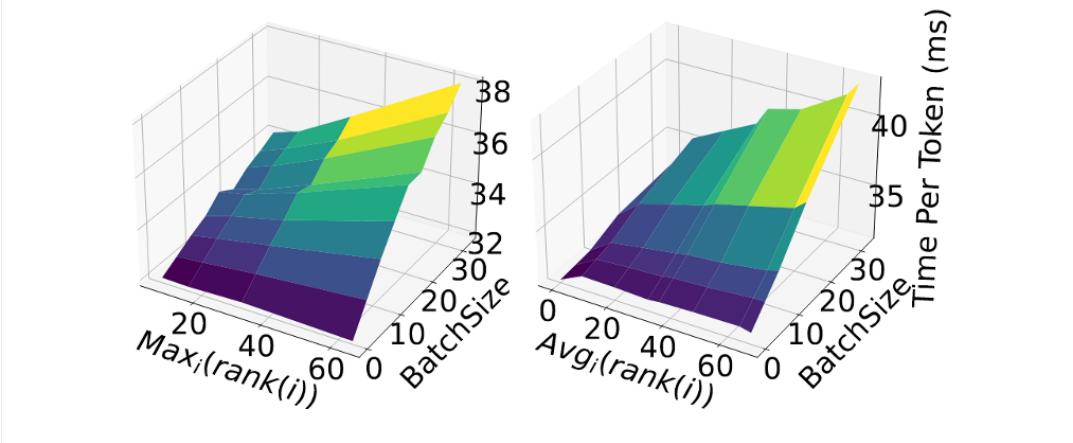
- 随着请求流量的增加，冷启动问题对系统性能的影响更加显著；
- 随着适配器秩的增加，加载到 GPU 的冷启动延迟也相应增加。

解决方法：

- 在GPU中预缓存所有adapter——显存占用大
- s-Lora采用predictive pre-fetching，由于请求具有高突发性，可能导致预测错误
- punica使用异步加载，但仍需要加载adpter的冷启动时间

Challenge2 异构 LoRA 服务的请求调度

对异构LoRA adapter计算（XAB）进行批处理的kernel设计

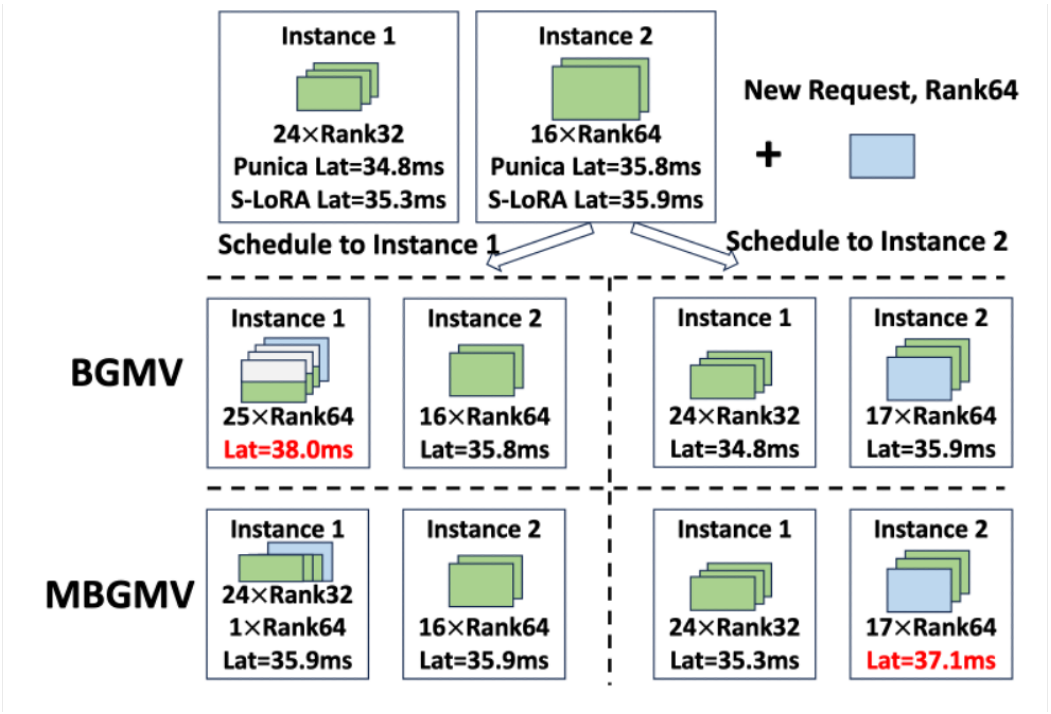


- BGMV（Punica）： 将小\rank的adapter填充到最大rank执行批处理

  性能（解码时间）取决于rank较大的adapter

- MBGMV（SLoRA）： 并不使用填充

  性能取决于平均rank



- 解码Latency SLO： 36 ms。
- 使用Punica的BGMV，将新请求调度到实例2满足SLO；
- 使用SLoRA 的MBGMV，将其调度到实例1可以满足 SLO。
- 调度算法需要考虑如何根据当前实例的负载情况（包括请求数量和适配器rank）来选择合适的服务器处理新请求，以确保满足设定的解码延迟 SLO。

系统架构

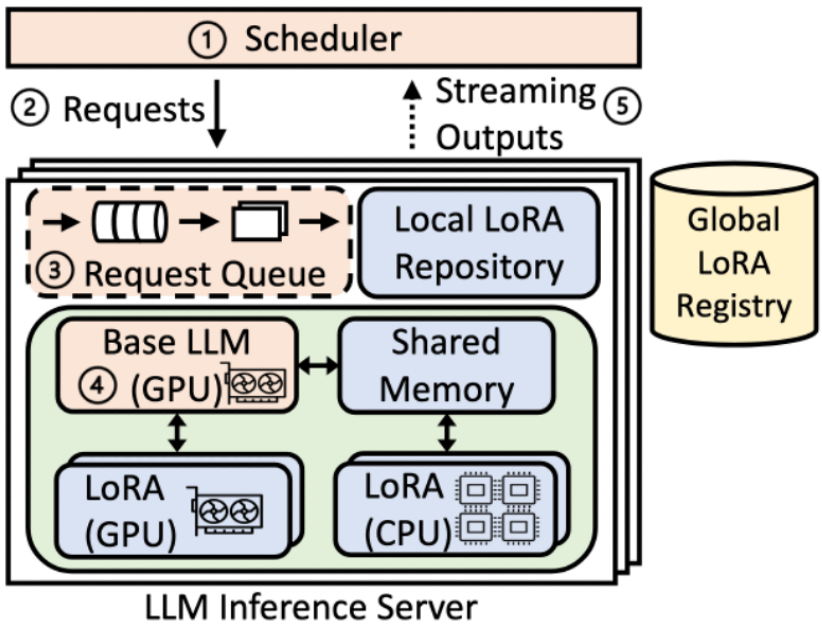


Figure 6: An architecture overview of CARASERVE.

- LLM inference server
- Scheduler：rank-aware scheduling algorithm

CPU-assisted LoRA serving

adapter先在CPU上进行prefill，同时将adapter加载到GPU，等到加载完毕之后，adapter继续在GPU上执行prefill和后续的decode——这样做的好处是将load adapter的时间和prefill的时间重叠起来，缓解了load adapter带来的冷启动开销。

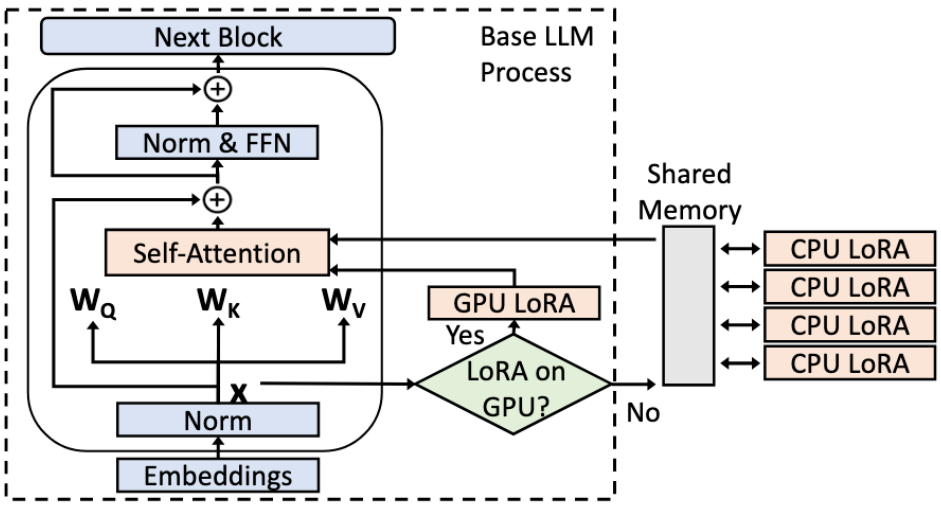


Figure 7: Illustration of coordinated LoRA computation on GPU and CPU per transformer block's attention layer.

对于输入x，当该请求的lora不在GPU中，就将x传到CPU中，执行xAB之后（prefill phase），同时从主机获取该lora adapter。当adapter加载完毕之后，就将CPU的结果传回CPU GPU LoRA就开始执行剩余的prefill和解码过程。

C1：adapter在CPU上的prefill结果和base model 在GPU上的prefill结果如何进行逐层同步?

on-GPU LLM 和 on-CPU lora这两者之间的同步

Solution：异步调用

Sync-free CPU LoRA invocation

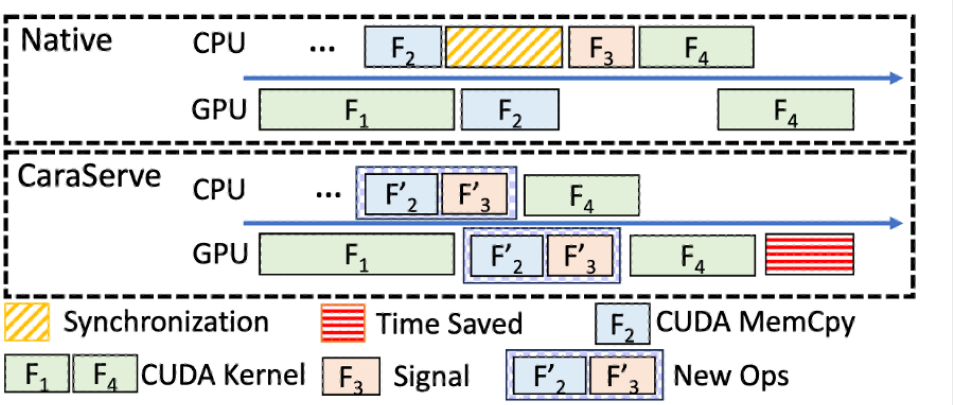


Figure 8: Execution timeline of Native LoRA Invocation and LoRA Invocation with CARASERVE's operator in base LLM process. CPU LoRA is ignored for simplicity.

native pytorch：CPU 执行主进程的调度

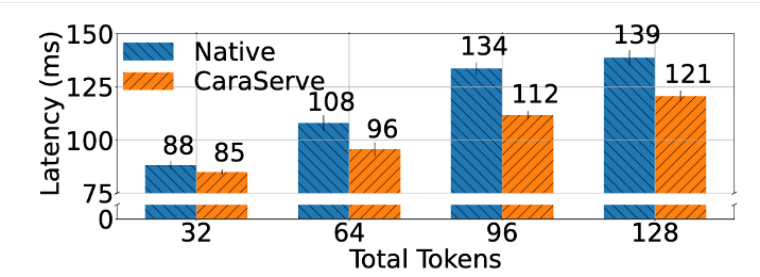
- cuda kernel F1计算xW
- cuda memcpy kernel F2将输入矩阵x从GPU传到CPU（图中的 CPU 上的 F2 表示 CPU 发出数据传输请求，而 GPU 上的 F2表示实际的数据传输操作）
- 显示同步确认x传输完成
- x传输完成之后，CPU使用signaling operator F3去通知CPU上的Lora进程执行xAB
- 通知之后启动下一个cuda kernel F4

原生pytorch调用CPU lora需要显式同步（确保F2在F3之前完成），阻止后续F4内核启动

caraserve：融合F2 F3到异步CUDA内核中 [F2',F3']

- 异步数据传输和信号融合：CaraServe 引入了一个定制的操作符，将数据传输（MemCpy）和信号传递操作融合为一个异步的 CUDA 内核 [F2',F3']。其中，F2'负责异步数据传输，将数据从 GPU 传到主机内存，F3' 则通过共享内存异步通知 CPU 上的 LoRA 进程，使其知道数据已经准备好，可以进行计算。
- 消除显式同步：在 Native 实现中，需要显式同步来确保数据传输完成后再发出信号，但在 CaraServe 中，这种同步被消除了。因为 CaraServe 将 F2' 和 F3' 融合成一个异步的 CUDA 内核，使得它们可以在没有同步的情况下同时执行。由于 CUDA 设备队列采用严格的先进先出（FIFO）执行顺序，可以保证数据的顺序和一致性，因此不需要额外的同步操作来确保数据的有效性。
- 无阻塞的后续内核执行：由于不再依赖显式同步，CaraServe 中的融合内核 [F2',F3'] 可以直接加入到 GPU 设备队列中，无需等待之前的内核 F1完成。因此，后续的 CUDA 内核（例如 F4）可以不被阻塞，立即排队执行，从而消除了同步带来的开销。

- 更高的效率：通过这种方法，CaraServe 避免了显式同步带来的 GPU 空闲时间，提升了 GPU 资源的利用率。实验结果表明，与 PyTorch 的 Native 实现相比，这种融合内核在每次预填充迭代的延迟上减少了 16%。



C2：Lora计算的频繁启动，进程间数据传输 导致高调用开销

通过共享内存实现base llm进程和CPU lora的进程间通信，GPU把输入矩阵x写入共享内存，cpu lora读取x并执行xAB，再将xAB的结果写回共享内存

C3：CPU的并行性有限，执行prefill可能产生瓶颈

profiling-guided lora parallelization scheme：跨多个CPU并行lora计算

对于输入x，维度是B * L * H，一个CPU Core最多可处理c个tokens，则分配`L/c`个core来计算xAB。

Rank-aware request scheduling

提高集群性能并满足用户的SLO

performance model

performance model预测lora adapter的解码延迟

对批请求中的rank异构性（影响因素有rank和batchsize）和服务性能（latency）之间的关系进行建模：线性回归模型 拟合度R2高达0.96

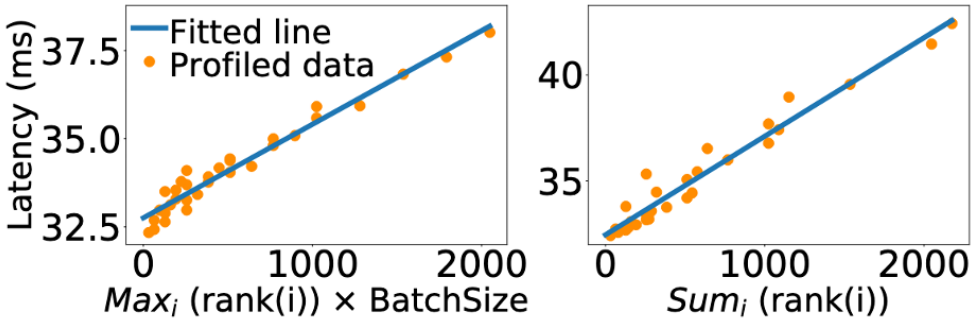


Figure 9: Performance models for BGMV (Left) and MBGMV (Right) kernels. Both linear regression models achieve a high coefficient of determination (R^2) of 0.96.

预测特定异构adapter批次的prefill和encoding延迟

调度策略Rank-aware request scheduling

基于性能模型 开发了Rank-aware request scheduling

当新请求到达时，调度程序会对拥有该请求Lora adapter的推理服务器进行评估，得到每个服务器的成本分数。

评估如果该新请求要在这个服务器中执行，对当前正在进行的请求带来的影响（额外延迟成本/slo违规）。选择具有最低成本分数的服务器执行新请求。

Algorithm 1: Rank-aware Scheduling Policy

Input: Performance models for *Prefill* and *Decoding*: $PrePerf(\cdot)$, $DecPerf(\cdot)$; average response length: avg_resp_len

```
while True do
    Request i arrives;
    candidates ← available LLM inference servers
    for instance in candidates do
        running_batch, queue = instance.GetStats()
        cost = CalcCost(i, running_batch, queue)
        requests = len(running_batch) + len(queue)
        instance.total_cost = cost * requests
    end
    best = min(candidates, key=lambda x: x.total_cost)
    best.serve(i)
end
Function CalcCost(req, running_batch, queue):
    exists = running_batch + queue
    # calculate additional prefilling time
    Δ_prefill = PrePerf(queue + req) - PrePerf(queue)
    # calculate additional decoding time per token
    Δ_decode = DecPerf(exists + req) - DecPerf(exists)
    cost_score = (Δ_prefill / avg_resp_len) + Δ_decode
    if DecPerf(exists + req) > SLO then
        cost_score += penalty_score
    end
    return cost_score
```

实验

实验设置

model：Llama2 7B/13B/70B

lora adapter应用在QKV权重矩阵

metrics：ttft、tpot、request latency

baselines：

- S-LORA
- OMDMD按需加载LORA adapter 存在冷启动开销
- CACHED 表示所有的LORA adapter都已经预缓存在GPU内存中，性能上限

除了S-LoRA（MBGMV）以外的baseline都配备BGMV内核

workload：

- Synthetic workload：到达LLM服务器的请求遵循泊松分布，可用于近似模拟调用
- Scaled production workload：采用MAF trace生成生产工作负载

End-to-End Performance on a Single GPU

tested：A10 GPU serving Llama2-7B

rank=64 请求到达符合泊松分布RPS=9

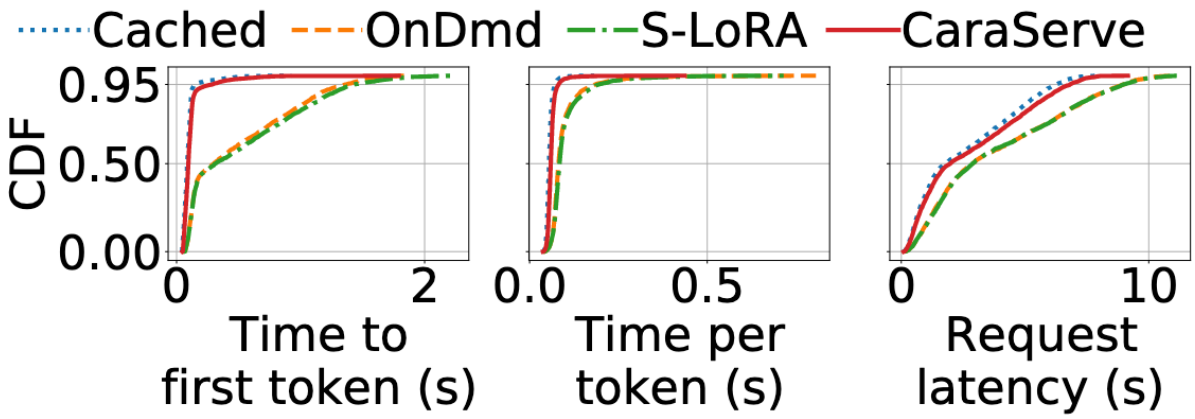


Figure 10: End-to-end results with Llama2-7B.

Caraserve的性能接近Cached

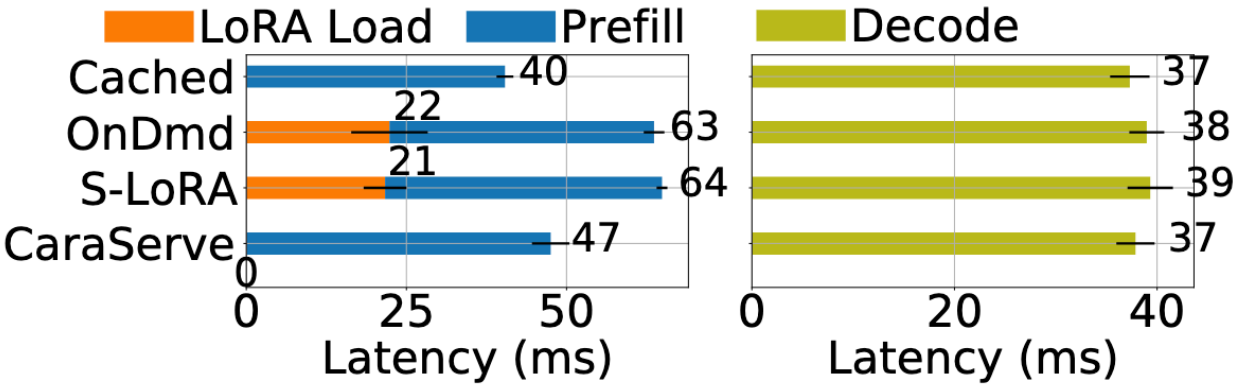


Figure 11: Prefill and decoding latency at LLM inference server. CARASERVE hides the LoRA adapter loading overhead by overlapping loading and CPU computation.

Caraserve通过重叠adapter loading 和 CPU Lora计算来隐藏lora load

S-Lora和ONDMD由于adapter loading较长的时间开销

Sensitivity Analysis

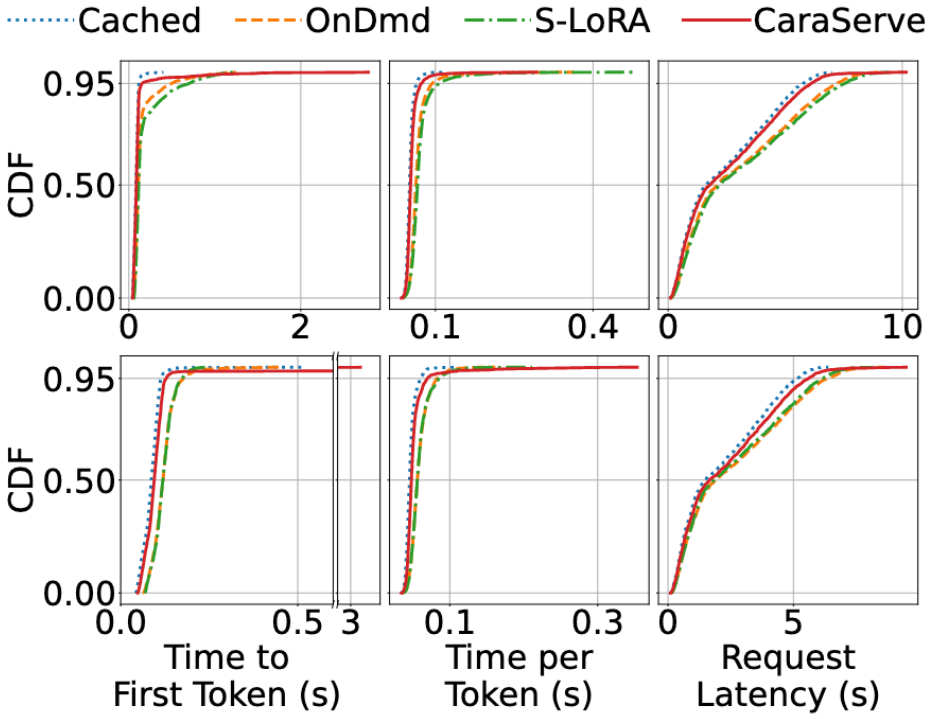


Figure 13: Sensitivity analysis of different Ranks and Traces. Top: $RPS = 9, rank = 32$; Bottom: $RPS = 6, rank = 64$.

- lora rank：rank越小 adapter loading的latency越短
- workload：决定了adapter loading的频率

End-to-End Performance on Multi-GPUs

tested：

Llama2-13B two A10 GPUs

Llama2-70B four A100 GPUs

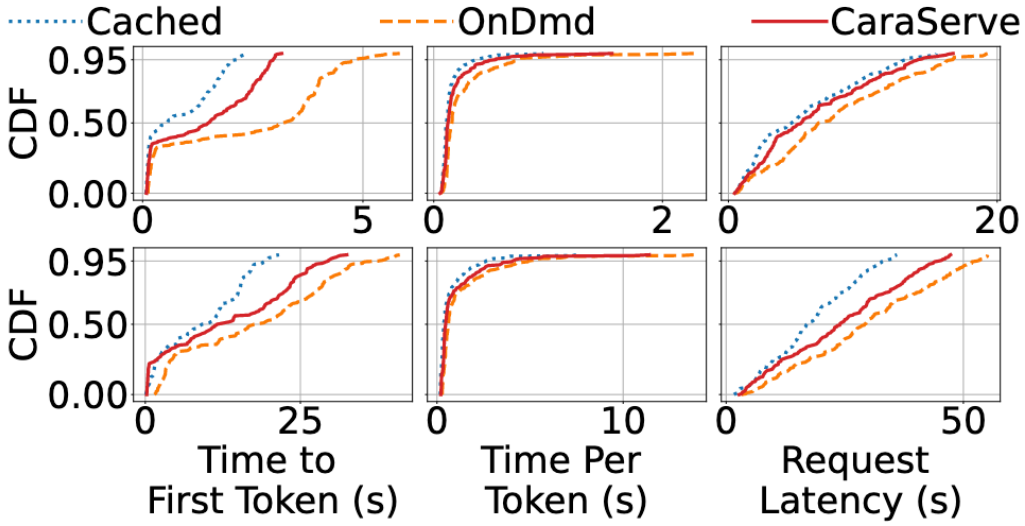


Figure 15: Evaluation on Llama2-13B (Top) and Llama2-70B (Bottom) models with $RPS = 6, rank = 64$.

相较于按需加载方法，CARASERVE 的性能显著提升：

- 在 Llama2-13B 和 Llama2-70B 模型的端到端请求延迟上，CARASERVE 分别实现了 20.2% 和 18.5% 的加速。
- 与 ONDMD 方法相比，CARASERVE 将冷启动开销减少了 50% 以上。

Scheduler

baseline:

- MOSTIDLE：优先选择最小工作负载的inference server
- FIRSTFIT：遵循the first-fit bin-packing strategy（Punica中也采用该策略）
- RANDOM：随机选择

tested：8×A10 GPU

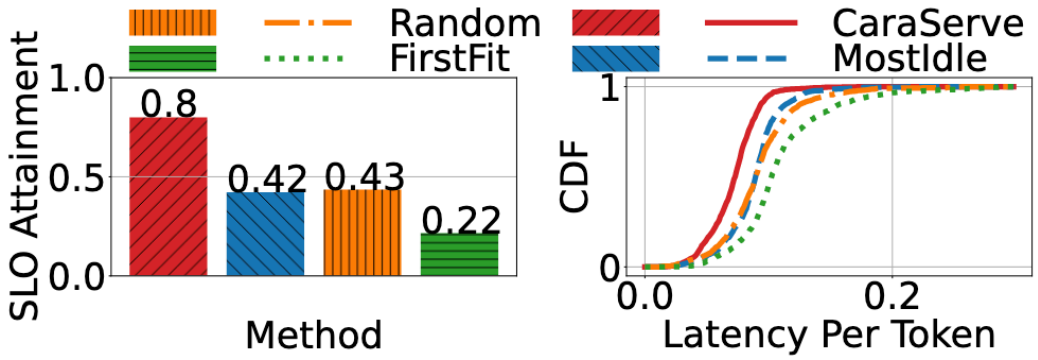


Figure 20: [**Testbed**] Scheduler performance on 8 instances (BGMV). Left: SLO attainment; Right: Time per token CDF.